Paper No. 43

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Ex parte WILLIAM VAN LOO, JOHN WATKINS, ROBERT GARNER, WILLIAM JOY, JOSEPH MORAN, WILLIAM SHANNON, and RAY CHENG

Appeal No. 95-4714
Application No. 08/046,476[1]

ON BRIEF

Before MARTIN, FLEMING, and BARRETT, Administrative Patent Judges.

MARTIN, Administrative Patent Judge.

DECISION ON APPEAL

This is an appeal from the final rejection of claims 13, 14, and 17-27, all of appellants' pending claims, as unpatentable under 35 U.S.C. § 103. We reverse and enter new grounds of rejection against claims 13 and 14.

We note that inasmuch as appellants' reply brief was refused

[1] Application for patent filed April 13, 1993, which is identified as a continuation of Application 07/603,248, filed October 24, 1990 (now abandoned), which is identified as a continuation of Application 07/104,280, filed October 2, 1987 (now abandoned).

entry by the examiner, it has not been considered.

The invention is a flushing system for a virtual cache memory in a computer workstation. Claim 13, which is one of the three independent claims on appeal (claims 17 and 23 are the others), reads as follows:

> 13. In a computer workstation operating in accordance with a shared, multi-user operating system having multiple concurrently active contexts and having a kernel wherein virtual addresses are assigned for each of a plurality of users, said workstation having a central processor and a cache data array coupled to an address bus, said cache data array including a plurality of cache blocks each one having a cache block address and an associated cache block tag, a system for completing a cache block flush operation, comprising:
>
> flush control logic means coupled to said address bus for controlling said cache block flush operation after receipt of a flush command from said central processor, said flush control logic means issuing a signal and asserting control of said address bus after receipt of said flush command, said flush control logic means retaining control of said address bus until said cache block flush operation is completed;
>
> reset means coupled to said flush control logic means for setting and resetting elements of said cache block tags;
>
> a plurality of cache flush control means disposed within said kernel of said shared, multi-user operating system and responsive to said signal issued by said flush control logic means, each of said flush control means comparing preselected portions of said cache block address and elements of said cache block tag for each of said plurality of cache blocks to different preselected criteria, said flush control means then flushing said cache blocks to said main memory if said

comparison results in a preselected relationship; and

a memory management unit coupled to said cache data array for reassigning virtual addresses to said plurality of cache blocks after said cache block flush operation is complete.

The examiner relies on the following references:

Freeman et al. (Freeman)	4,677,546	Jun. 30, 1987
Stiffler et al. (Stiffler)	4,819,154	Apr. 4, 1989

Claims 13 and 14 stand rejected under § 103 as unpatentable over Stiffler, while claims 17-27 stand rejected under § 103 as unpatentable over Stiffler in view of Freeman.

New § 112 Rejections Entered Pursuant to 37 C.F.R. § 1.196(b)

For the following reasons, claims 13 and 14 are hereby rejected under 35 U.S.C. § 112, second paragraph, for being indefinite, i.e., for failing to particularly point out and distinctly claim what appellants regard as their invention. In addition, these claims are rejected under the written description requirement of the first paragraph of § 112 as containing new matter. Both of these new grounds of rejection are based on the paragraph in claim 13 that begins "a plurality of cache flush control means." Before considering that paragraph, we will read the other claim recitations onto appellants' specification and drawings.

Beginning with the preamble, the claimed "computer workstation" reads on the elements shown in Figure 1, which includes the claimed central processor (CPU 11), which generates a virtual address of A bits in size which uniquely identifies bytes of instructions or data within a given virtual context (Spec. at 5, lines 13-15). The operating system, which is common to all processes or contexts, lies within a common region at the top of the 2^A bytes virtual address space for each context (Spec. at 10, lines 1-16). The specification indicates (at 1, lines 2-5) that the operating system is "multi-user" and has "multiple concurrently active contexts," as required by the claim. The claim's recitation that the operating system "ha[s] a kernel wherein virtual addresses are assigned for each of a plurality of users" finds support in the paragraph bridging pages 17 and 18. As also required by the preamble, the workstation has a cache data array (19), which includes a plurality of cache blocks, each having a virtual cache block address and an associated block tag (which is stored in cache tag array 23). Although not required by the claim, we note that context register 32 contains C virtual address bits which identify the currently running context or process (Spec. at 10, lines 1-3).

The paragraph that begins "flush control logic means" reads

on the disclosure in the following manner. The CPU issues a Flush command that is described as follows in the paragraph bridging pages 28-29:

> The Flush command is issued by the CPU in Control Space (identified by Function Code bits FC(2:0)=0x3). Within Control Space, the four high order address bits A(31:28)=0xA indicate the Flush command. The address field A(27:0) for the command correspond to the 28 bit virtual address field for data accesses. The Flush command data bits D(1:0) encode the type of flush.

The types of flushes include context, page, and segment (Spec. at 28, lines 15-17).

The Flush command address and data bits are provided as input signals to the cache flush block diagram shown in Figure 11, which represents the operation of cache flush logic 33 of Figure 1 (Spec. at 28, lines 3-4). This logic includes an AND gate 48, flip-flops 49, flush address register 52, incrementer 50, AND gates 55, and OR gate 58 (Spec. at 28, lines 7-9). The specification explains (at 29, lines 6-9) that "[a]fter the Flush command is decoded [by AND gate 48], the address field A(27:9) is latched [in flush address register 52] together with the type of flush [in flip-flops 49]. A Bus Request signal is asserted to the CPU to obtain bus mastership." The CPU then issues a Bus Grant signal to the flush control logic, which retains control of the cache address bus until after the last of the cache blocks

has been checked (Spec. at 29, lines 20-22).

During a flush operation, all thirty-two cache blocks in the cache data array are addressed in sequence using the virtual address bits A(8:4) generated by 5-bit incrementer 50 (Spec. at 29, lines 10-15). As is apparent from Figure 11, an addressed cache block is flushed (i.e., a Flush Match signal is generated by OR gate 58 of Fig. 11) only if the Context Flush, Page Flush, or Segment Flush signal stored in one of flip-flops 49 is applied to one input of an AND gate 55 at the same time that a Context Match, Page Match, or Segment Match signal is applied to the other input of that AND gate. These Match signals are produced by the circuitry shown in Figure 12 (Spec. at 28, lines 18-20), which compares the virtual address A(27:4) of the addressed cache block (including the cache block address and the associated address information in the cache tag array) with the virtual addresses to be flushed, compares the context identification bits CX(2:0) of the addressed cache block with the context identification bits of the virtual addresses to be flushed, and examines the Valid (V), Modified (M), and Protection (P) bits stored in the cache tag array for the addressed block (Spec. at 15-17).

The foregoing elements clearly perform the functions

required of the "flush control logic means."

The next paragraph, which calls for "reset means coupled to said flush control logic means for setting and resetting elements of said cache blocks tags," reads on the step of invalidating the Valid bits in the cache tags as part of the flushing process (Spec. at 11, lines 5-7).

Skipping over the next paragraph for a moment, the last paragraph requires a memory management unit coupled to the cache array for reassigning virtual addresses to the plurality of cache blocks after the flush operation is complete. This paragraph accurately describes the function of appellants' MMU 27 (Spec. at 19, lines 10-18).

The claim paragraph which is the basis for the § 112 rejections reads as follows:

> a plurality of cache flush control means <u>disposed within said kernel</u> of said shared, multi-user operating system and responsive to said signal issued by said flush control logic means, each of said flush control means comparing preselected portions of said cache block address and elements of said cache block tag for each of said plurality of cache blocks to different preselected criteria, said flush control means then flushing said cache blocks to said main memory if said comparison results in a preselected relationship. [Emphasis added.]

The problem is that the claimed functions are disclosed as being performed by hardware rather than by the operating system

"kernel."[2] As explained above, the role of the CPU in the flushing operation is limited to issuing a Flush command and responding to a Bus Request signal by issuing a Bus Grant signal. The claimed comparison function is performed by the circuitry depicted in Figure 12, which provides Context Match, Page Match, and Segment Match signals to the circuitry of Figure 11, which issues a Flush Match signal when the requisite conditions have been satisfied. Furthermore, the recitation that the kernel is involved in the comparison and flushing functions contradicts the "flush control logic means" paragraph, which specifies that the flush control logic means maintains control of the address bus from the time it receives a flush command from the central processor until the cache block flush operation has been

[2] Because the specification does not provide a definition of "kernel," it is given its broadest reasonable interpretation consistent with appellants' disclosure. In re Zletz, 893 F.2d 319, 321, 13 USPQ2d 1320, 1322 (Fed. Cir. 1989). Neither appellant nor the examiner has provided a definition of this term. We note it is described as follows in A. Silberschatz & P. Galvin, Operating System Concepts 5 (4th ed. 1994) (copy enclosed): "There is . . . no universally accepted definition of what is part of the operating system and what is not. . . . [T]he operating system is the one program running at all times on the computer (usually called the kernel), with all else being applications programs" (emphasis in original).

completed.[3]

For the foregoing reasons, claim 13 is indefinite and in violation of the second paragraph of § 112. Furthermore, because the "disposed within the kernel" limitation was not present in the application as filed, claim 13 also violates the written description requirement of the first paragraph of § 112. Dependent claim 14 does not cure these violations and is therefore rejected for the same reasons.

Inasmuch as claims 13 and 14 are indefinite, it is not possible to apply the prior art to these claims in deciding patentability without disregarding portions of the express wording of the claims and thus resorting to speculation and conjecture as to the particular invention defined therein. For this reason, we will not sustain the examiner's § 103 rejection of these claims based on Stiffler. See In re Steele, 305 F.2d 859, 862-63, 134 USPQ 292, 295 (CCPA 1962).

Nevertheless, we have considered Stiffler to determine the extent to which it satisfies the claim limitations other than those in the indefinite paragraph and conclude that it does. Beginning with the preamble, Stiffler discloses a computer system

[3] Also, there is no antecedent basis for "said main memory."

(Fig. 1) which includes a number of processing elements (100, 105, 110) which are connected to a plurality of main system memory elements (165, 170, 175, 186) via processor buses (115, 116), master interfaces (120, 125), system buses (130, 131), slave interfaces (135, 140, 145, 150), and memory buses (155, 156, 160, 161). As shown in Figure 2, each processing element contains a microprocessor unit (MPU) 210, a cache memory 250 of the non-write through type (col. 8, lines 4-5), a block status memory 255, a memory management unit (MMU) 200, an internal sequence controller 240, and an external control sequencer 245. The MPU is "a conventional data processing device capable of executing both user application programs and supervisor programs which control and coordinate the operation of the associated processor" (col. 4, lines 8-12). Accordingly, the cache memory contains fixed supervisor code and data, overlayable supervisory code and data, and user code and data (see Fig. 5). We agree with the examiner that one of Stiffler's processing elements (e.g., 100) can be considered to be a "computer workstation" in the sense of claim 13.

Claim 13 additionally requires that the operating system be of the "multi-user" type and "have multiple concurrently active contexts." The examiner contends (Answer at secs. 9b and 11d),

and we agree, that it would have been obvious to have a plurality of different users operate a plurality of Stiffler's workstations (i.e., processing elements) at the same time. Even though each workstation runs only one context at a time (see col. 9, lines 27-31), the result will be a multi-user operating system that concurrently runs plural contexts.

The preamble's requirement that the operating system have a "kernel wherein virtual addresses are assigned for each of a plurality of users" also appears to be satisfied by Stiffler's operating system, which causes the MPU (210) in each processing element to produce a 16 megabyte range of virtual addresses between 000000 and FFFFFF virtual addresses within a specified range (col. 7, lines 3-8; see Figs. 5 and 6). The remaining preamble limitations are also satisfied as follows: the claimed cache data array (250 in Fig. 2) includes a plurality of cache blocks each having a cache block address and an associated cache block tag (which is stored in block status memory 255 in Fig. 2).

The foregoing elements have different reference numerals in the detailed workstation block diagram formed by Figures 7 (sheets 1 and 2) and 8. The central processor is MPU 702, the cache data array is cache RAM 738, and the block status memory is labeled 736. These figures also depict the elements of

Stiffler's flush control circuitry, which he describes as "special purpose hardware" (col. 17, lines 56-57) and which flushes one or more blocks from the cache data array in response to a context switch or an overflow situation (col. 2, lines 8-11). The flush control circuitry is controlled by the internal sequence controller (unnumbered in Fig. 7 but identified by numeral 700 in the specification), which causes the processing element to assume one of eight operating states (col. 16, lines 4-5). The flushing operation that occurs during a context switch is described in general at column 17, line 50 to column 18, line 3 and in detail at column 32, line 54 to column 36, line 49. The operation begins with MPU 702 commanding a between-limits flush (col. 17, lines 59-60). This command is in the form of an address signal having the format shown in line G1 of Figure 9 (col. 33, lines 63-65). Bits 7-16 identify the cache block at which to begin the flush operation (col. 33, lines 67-67). Bits 1-3 are control bits, of which bit CXT is set to zero to indicate that the flush is being done as part of a context switch (col. 34, lines 6-13). All of these bits (1-3 and 7-16) are placed on the local address bus 730 (col. 33, lines 60-68). The MPU also generates, on local data bus 732, a termination address having the format shown in line G2 of Figure 9, of which bits 7-

16 constitute the termination address (col. 34, lines 18-27). In response to issuance of the flush command signal by MPU 702, flush control is assumed by special purpose hardware, which under the control of the internal sequence controller carries out the flushing operation independently of direct control by the MPU, retaining control until the last block has been flushed (col. 17, lines 54-59; col. 32, lines 62-66). At the start of the flushing operation, bits 6-14 of the start address are loaded into counter 718 (col. 60-63), which apparently is incremented each time a cache block has been processed. Flush operation is terminated when the address stored in the counter equals the termination address appearing on local data bus 730 (col. 34, lines 54-57). After the flush operation has been completed, the internal sequence controller issues an acknowledge signal (ACK) to the MPU (col. 36, lines 45-49). It is readily apparent that the above-described special purpose circuitry functions as "flush control logic means coupled to said address bus for controlling said cache block flush operation after receipt of a flush command from the central processor," as required by claim 13. The question is whether it also satisfies the requirement that the control logic means "issu[e] a signal and assert[] control of said address bus after receipt of said flush command" and "retain[] control of

said address bus until said cache block flush operation is completed." We believe it does, when the recited "address bus" is read on the address bus (737) that is directly connected to the input of cache RAM 738. At the start of the flushing operation, the internal sequence controller, acting through multiplexers 720 and 726, causes bits 5-14 on this address bus to be controlled by counter 718 (col. 34, lines 34-36). This counter thus controls addressing of the cache until the flush operation is complete.

The claimed "reset means" is also satisfied. When the flush is being done as part of a context switch, the "valid" bit in the block status memory will be cleared (col. 34, lines 8-10).

Skipping over the next paragraph, which we have determined is indefinite, the claimed "memory management unit" reads on MMU 200 in Figure 23, the details of which are shown in Figure 8 (col. 3, lines 49-51).

The § 103 Rejection of Claims 17-27

Claims 17-27 stand rejected under § 103 as unpatentable over Stiffler in view of Freeman. Claim 17 reads as follows:

> 17. A computer system with cache flushing comprising:
>
> a central processing unit operating in accordance

with a shared, multi-user operating system having multiple concurrently active contexts and a kernel;

main memory;

a memory management unit, coupled to said processor and said main memory, for translating an address in virtual space into a corresponding address in physical space;

a virtual cache data array, coupled to said central processing unit, for storing a first plurality of blocks of data;

a virtual cache tag array, coupled to said virtual cache data array and said central processing unit, for storing a plurality of tag array elements wherein, each tag array element corresponds to a particular block of data stored in said virtual cache data array and further includes: a validity bit, a modification bit, a protection bit, a write allowed bit, a plurality of virtual address field bits, and a plurality of context bits;

cache hit logic means, coupled to said processor and said virtual cache tag array, for determining whether accesses from said central processing unit result in a cache hit or a cache miss;

cache flush logic means, coupled to said central processing unit and said cache hit logic means, for directing the flushing of said virtual cache data array;

wherein said central processing unit includes means, disposed within the kernel of said shared, multi-user operating system, for coupling a context match flush command comprising a plurality of context identifier bits, to said cache flush logic means and said virtual cache tag array, such that in response to said context match flush command, said cache flush logic means flushes a first block of data from said virtual cache data array in the event that:

the protection bit in the tag array element corresponding to said first block of data is in a first predesignated state; and,

the plurality of context bits in the tag array element corresponding to said first block of data match said plurality of context identifier bits.

The preamble is satisfied by Stiffler for the reasons given above in the discussion of the preamble of claim 13. The elements in the body of the claim correspond as follows to the circuitry in Stiffler's Figures 1, 2, 7 and 8:

(a) "main memory" - elements 165, 170, 175, and 184.

(b) "memory management unit" - MMU 210 in Fig. 2.

(c) "virtual cache data array" - cache memory 250 in Fig. 2.

(d) "virtual cache tag array . . . for storing a plurality of cache tag element" - block status memory 255 in Fig. 2.

(e) "each tag array element including . . ."

(1) "a validity bit" - "valid" bit (col. 9, lines 27-31).

(2) "a modification bit" - "dirty" bit (col. 9, lines 31-35).

(3) "a protection bit" - discussed below.

(4) "a write allowed bit" - discussed below.

(5) "a plurality of virtual address field bits" - the address "label" stored in the block status memory (col. 13, lines 36-42).

(6) "a plurality of context bits" - discussed below.

(f) "cache hit logic means" - internal sequence controller 700 in Fig. 8 (col. 22, lines 37-40).

(g) "cache flush logic means" - the circuitry of Figs. 7 and 8.

(h) "a context match flush command comprising a plurality of context identifier bits" - discussed below.

(I) the cache flush logic means causes flushing if the protection bit is in a first designated state and if the context bits in the tag array elements match the context bits in the context match flush command - discussed below.

Regarding the "context" limitations, the examiner contends (Answer at sec. 11g) that

> Stiffler by necessity must identify which context is currently active, since context switching is provided for, so Appellant's claimed "context identification registers" [sic, "context identification register"[4]] is not patentably distinguishing, since it is well known in the art for registers to store identification data.

Even assuming for the sake of argument that the examiner's reasoning is correct, the only purpose served by such a register would be to keep the MPU apprised of the identity of the context that is currently running. The examiner has not explained, and it is not apparent to us, why this reasoning would have led the artisan to additionally include context identification bits in the block status register and in the flush command signal, as

[4] See dependent claims 21, 22, 26 and 27.

required by the claim. In fact, such context identification information would appear to be superfluous in Stiffler's system, wherein only one context runs at a time on a workstation and wherein the "valid" bit in the block status memory "indicates whether the contents of the associated block are valid in the present context (associated with the program presently running in MPU 210)" (col. 9, lines 27-31). The examiner also has not explained, and it is not apparent to us, why Freeman obviates the foregoing shortcoming of Stiffler. Consequently, we cannot sustain the § 103 rejection of claim 17 or its dependent claims 18-22 as unpatentable over Stiffler in view of Freeman. For the same reason, we cannot sustain the rejection of independent claim 23 and its dependent claims 24-27 over those references.

As a result, we do not reach the question of whether the examiner it is correct to argue that it would have been obvious to modify Stiffler so as to employ protection bits of the types disclosed in Freeman order to achieve increased data integrity and security (Answer at sec. 9), thereby satisfying claim 17's requirement that the tag element include a "write allowed" bit and a "protection" bit.

* * *

This decision contains new grounds of rejection entered

pursuant to 37 C.F.R. § 1.196(b) (amended effective Dec. 1, 1997, by final rule notice, 62 Fed. Reg. 53,131, 53,197 (Oct. 10, 1997), 1203 Off. Gaz. Pat. & Trademark Office 63, 122 (Oct. 21, 1997)). 37 CFR § 1.196(b) provides that, "A new ground of rejection shall not be considered final for purposes of judicial review."

37 CFR § 1.196(b) also provides that the appellants, WITHIN TWO MONTHS FROM THE DATE OF THE DECISION, must exercise one of the following two options with respect to the new grounds of rejection to avoid termination of proceedings (§ 1.197(c)) as to the rejected claims:

> (1) Submit an appropriate amendment of the claims so rejected or a showing of facts relating to the claims so rejected, or both, and have the matter reconsidered by the examiner, in which event the application will be remanded to the examiner. . . .
>
> (2) Request that the application be reheard under § 1.197(b) by the Board of Patent Appeals and Interferences upon the same record. . . .

No time period for taking any subsequent action in connection with this appeal may be extended under 37 CFR § 1.136(a).

REVERSED; 37 C.F.R. § 1.196(b)

JOHN C. MARTIN Administrative Patent Judge	)	
LEE E. BARRETT Administrative Patent Judge	)	BOARD OF PATENT APPEALS AND INTERFERENCES
MICHAEL R. FLEMING Administrative Patent Judge	)	

Blakely, Sokoloff, Taylor and
Zafman
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, CA 90025

Enclosure: A. Silberschatz & P. Galvin, Operating System Concepts 3-6 (4th ed. 1994)